The Context Interchange Network Prototype

Adil Daruwala
Cheng Goh
Scott Hofmeister
Karim Hussein
Stuart Madnick
Michael Siegel

WP #3797 February 1995

The Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

The Context Interchange Network Prototype*

Adil Daruwala
Cheng Goh
Scott Hofmeister
Karim Hussein
Stuart Madnick
Michael Siegel

Sloan WP #3797 CISL WP #95-01 IFSRC WP #308-95
February 1995

The Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

* submitted to Proceedings of the VLDB

services that will be offered by the CIN. In Section 3, we examine the architecture of CIN and describe the major components and their functionality. We consider its use in providing semantic interoperability as well as its application to context management. In Section 4, we present details of the CIN implementation focusing on the conflict detection, and conversion-based query optimization. In Section 5, we present our strategy for introducing this technology into existing applications as a "middleware" service. In Section 6, we examine related research areas and future plans for the implementation.

1.1 The Need for Context Interchange

The need for the context interchange network is exemplified by the traditional approach to handling disparate context as shown in Figure 1. Numerous sources and receivers may exist on a network but both the stored data and the receivers (e.g., applications) have implicit preconceived assumptions about the meaning of the data. Traditionally conversions routines are introduced either on the source side (e.g., source S_1 provides conversion x_{12} for information sent to receiver R_2) or on the receiver side (e.g., receiver R_1 provides conversion x_{21} for information from source S_1) and in some instances on both sides[1]. Clearly this approach scales poorly since the total number of conversions is proportional to the product of the number of sources and the number of receivers (i.e. n^2). In addition there is no facility to explain differences in the meaning of data from different sources as knowledge about the meaning of data is embedded in conversion code.

The CIN approach, as depicted in Figure 2, allows for the explicit representation of source and receiver context knowledge and uses this information along with context mediation services to automatically determine the semantic conflicts between sources and receivers and to apply the appropriate conversions. As will be discussed later this approach is scalable and evolutionary with changes in context knowledge.

[1] Conversion x_{12} may have existed but the requirements of R_1 may have changed. Thus, x_{21} translates the output of x_{12} into the new form.

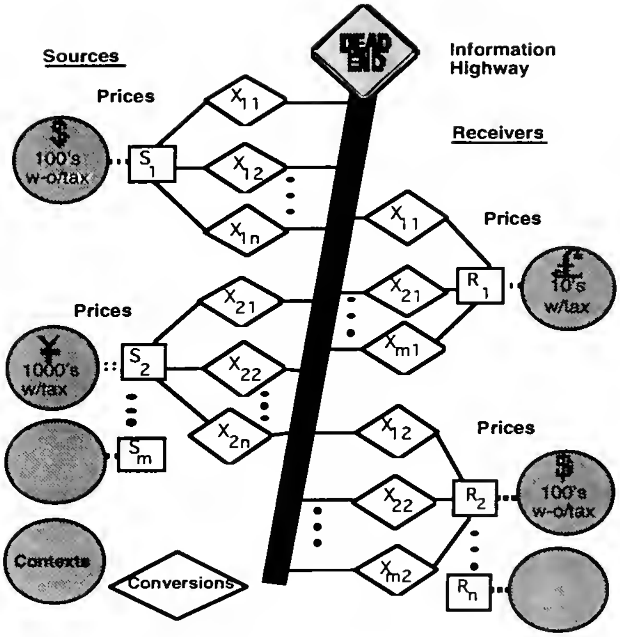


Figure 1: Traditional Approach to Handling Disparate Contexts

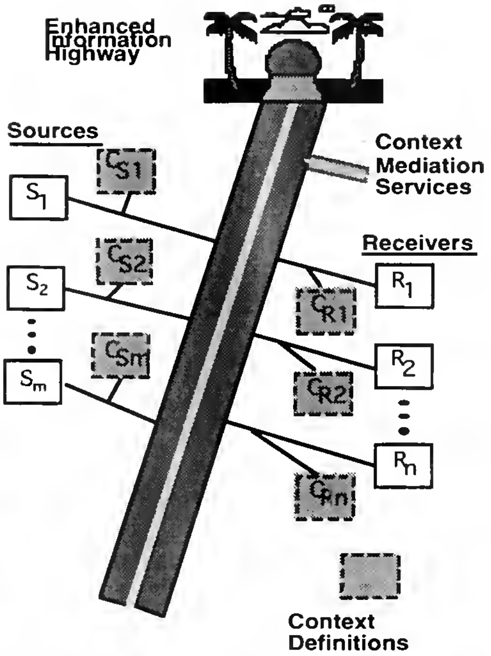


Figure 2: Context Interchange Approach

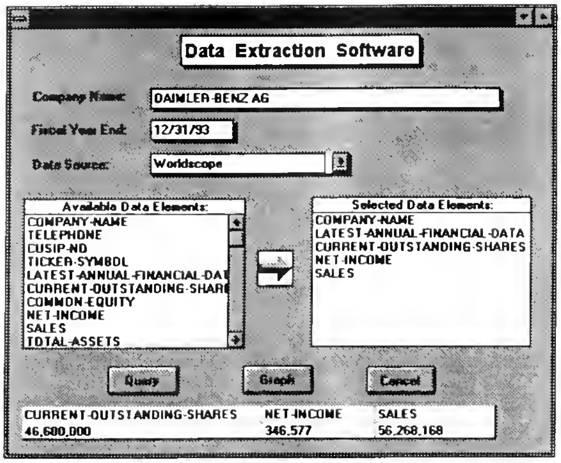
2 An Example Scenario Using Context Interchange Network

As an example of this approach we describe the following scenario in which an executive of an investment bank, is attempting to compile historical information on Daimler-Benz for a meeting of the bank's investment advisory committee.

2.1 Traditional Approach

She starts up the Data Extraction Software, a Powerbuilder application with access to several relational databases, and proceeds to enter DAIMLER-BENZ AG as the company name. She needs to compile historical data for the past fiscal year, thus she enters 12/31/93 as the Fiscal Year End date and selects the Worldscope financial data service since she is most familiar with it, and proceeds to select the NET-INCOME, SALES, and CURRENT-OUTSTANDING-SHARES data fields[2]. She notices that the COMPANY-NAME and the LATEST-ANNUAL-FINANCIAL-DATA fields have been pre-selected by the application, and remembers that the application uses them for graphing the selected data elements. She then clicks the Query button to extract the necessary information (Figure 3). She makes a note of the extracted data as reported in the lower half of the Data Extraction Software window.

[2] Note that the typeface convention that we will use in this paper is that attribute names will be in all UPPERCASE, concepts will be represented in SMALL-CAPS COURIER, meta-attributes will be in plain courier, values of meta-attributes will be in [bracketed].



Company Name: DAIMLER-BENZ AG

Fiscal Year End: 12/31/93

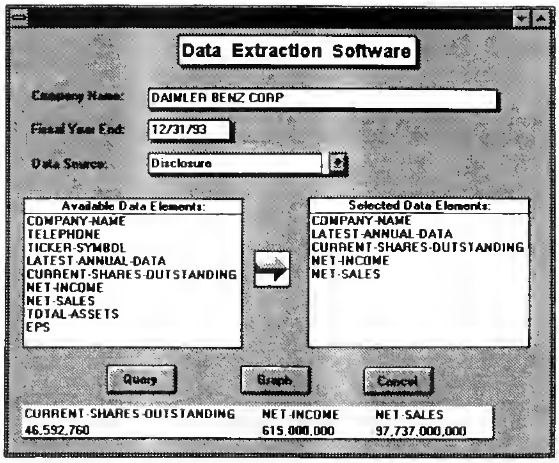
Data Source: Worldscope

Available Data Elements:
COMPANY-NAME
TELEPHONE
CUSIP-NO
TICKER-SYMBOL
LATEST-ANNUAL-FINANCIAL-DAT
CURRENT-OUTSTANDING-SHARE
COMMON-EQUITY
NET-INCOME
SALES
TOTAL-ASSETS

Selected Data Elements:
COMPANY-NAME
LATEST-ANNUAL-FINANCIAL-DATA
CURRENT-OUTSTANDING-SHARES
NET-INCOME
SALES

```
CURRENT-OUTSTANDING-SHARES     NET-INCOME     SALES
46,600,000                     346,577        56,268,168
```

Figure 3: Data Extraction Software - Worldscope



Company Name: DAIMLER BENZ CORP

Fiscal Year End: 12/31/93

Data Source: Disclosure

Available Data Elements:
COMPANY-NAME
TELEPHONE
TICKER-SYMBOL
LATEST-ANNUAL-DATA
CURRENT-SHARES-OUTSTANDING
NET-INCOME
NET-SALES
TOTAL-ASSETS
EPS

Selected Data Elements:
COMPANY-NAME
LATEST-ANNUAL-DATA
CURRENT-SHARES-OUTSTANDING
NET-INCOME
NET-SALES

```
CURRENT-SHARES-OUTSTANDING     NET-INCOME     NET-SALES
46,592,760                     615,000,000    97,737,000,000
```

Figure 4: Data Extraction Software - Disclosure

To verify the authenticity of the data, she repeats the process to extract the same data elements from the Disclosure financial data service. The Data Extraction Software application reports that Disclosure does not recognize DAIMLER-BENZ AG as a valid company name. She spends a few minutes trying various combinations of the company name before finding the right combination - DAIMLER BENZ CORP. She also notices that the available field names are different from those supplied by the Worldscope. She selects the NET-INCOME, NET-SALES, and CURRENT-SHARES-OUTSTANDING data fields and proceeds to extract the necessary information (Figure 4). She again makes a note of the extracted data as reported in the lower half of the Data Extraction Software window.

Upon extracting the data, she immediately notices a large difference in the income and sales data from the Worldscope and Disclosure data service. She is unsure of the cause for this difference - the discrepancy could be due to differences in reporting units, currencies, or other contextual differences between Worldscope and Disclosure data. She tries to call the Customer Service for the two data services but finds them closed for the weekend.

2.2 Context Interchange Network Approach

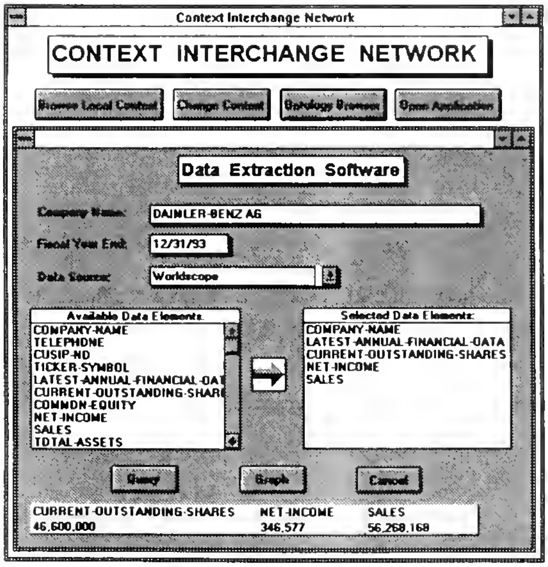
Being unsure of the validity of the extracted data, she decides to use the Context Interchange Network (CIN) to resolve the context problems that she has encountered. After she starts up the CIN and identifies herself which indicates that her context correspond to the Worldscope data context, she proceeds to open the same Data Extraction Software application from within the CIN.

To verify the functionality of the Context Interchange Network, she repeats the exercise in extracting the data from the Worldscope data service for the fiscal year ending 12/31/93. She notices that the data reported by Worldscope through CIN (Figure 5) is identical to that reported directly by the Worldscope data service earlier (Figure 3).

She then changes the data source to Disclosure. She immediately notices that though she has selected Disclosure as the data source, the application accepts the Worldscope naming convention DAIMLER-BENZ AG as the company name, and that it reports the data fields names in Worldscope's context. She selects the NET-INCOME, SALES, and CURRENT-OUTSTANDING-SHARES fields, and proceeds to extract the data (Figure 6).

She notices that the data now reported by Disclosure (Figure 6) is very different from that reported directly by Disclosure earlier (Figure 4). She also notices that the data now reported by Disclosure (Figure 6) relates closely to the data extracted from the Worldscope data service (Figures 3 and 5).

The use of the CIN allowed the analyst to examine information in a common, familiar context (i.e., Worldscope) and to meaningfully compare information from other contexts (i.e., Disclosure). This was all possible without changing the existing application (i.e., Data Extraction Software). In the remainder of this paper, we describe how the CIN makes this possible.



Company Name: DAIMLER-BENZ AG

Fiscal Year End: 12/31/93

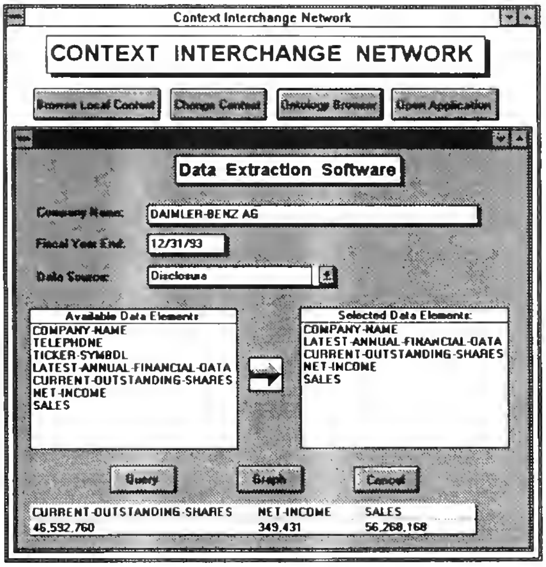
Data Source: Worldscope

Available Data Elements:
COMPANY-NAME
TELEPHONE
CUSIP-NO
TICKER-SYMBOL
LATEST-ANNUAL-FINANCIAL-DAT
CURRENT-OUTSTANDING-SHARE
COMMON-EQUITY
NET-INCOME
SALES
TOTAL-ASSETS

Selected Data Elements:
COMPANY-NAME
LATEST-ANNUAL-FINANCIAL-DATA
CURRENT-OUTSTANDING-SHARES
NET-INCOME
SALES

```
CURRENT-OUTSTANDING-SHARES   NET-INCOME   SALES
46,600,000                   346,577      56,268,168
```

Figure 5: Context Interchange Network -- Worldscope Data



Company Name: DAIMLER-BENZ AG

Fiscal Year End: 12/31/93

Data Source: Disclosure

Available Data Elements:
COMPANY-NAME
TELEPHONE
TICKER-SYMBOL
LATEST-ANNUAL-FINANCIAL-DATA
CURRENT-OUTSTANDING-SHARES
NET-INCOME
SALES

Selected Data Elements:
COMPANY-NAME
LATEST-ANNUAL-FINANCIAL-DATA
CURRENT-OUTSTANDING-SHARES
NET-INCOME
SALES

```
CURRENT-OUTSTANDING-SHARES   NET-INCOME   SALES
46,592,760                   349,431      56,268,168
```

Figure 6: Context Interchange Network -- Disclosure Data

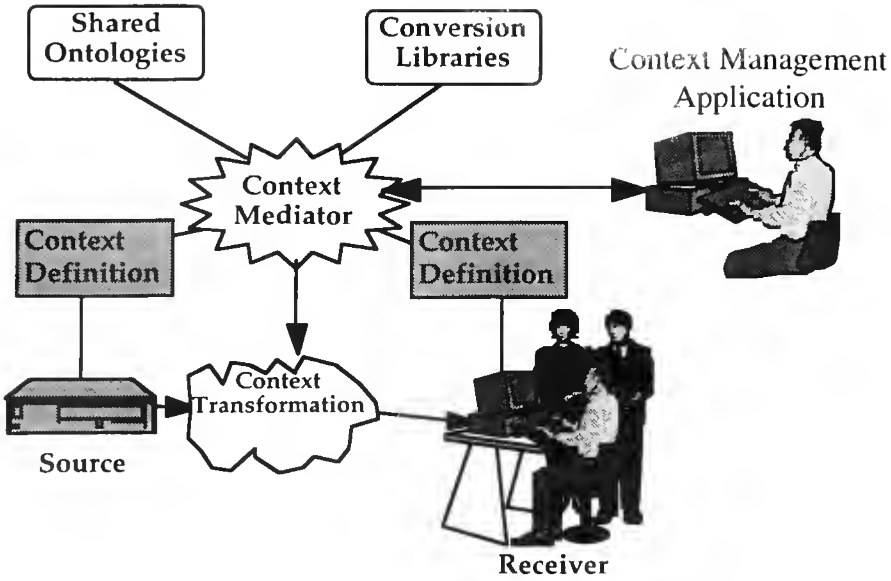


Figure 7: Context Interchange Network Architecture

3. The Context Interchange Network Architecture

In this section we present a high level view of the architecture of the current implementation as shown in Figure 7, by examining the components and functionality of the system for a single source and single receiver. It is important to note that there are two major uses of this network. First, is to guarantee semantic interoperability for new or existing receiver applications as exemplified by the example in Section 2. The second use is for the management of contexts, as described below. For simplicity, we refer to any source (e.g., database, data feed, application program output) or receiver (e.g., application, consolidated database) as a *node* in the CIN.

3.1 Context Mediation

In the Context Interchange Network every data source must specify a context definition which describes the kinds of data that it can supply (e.g., COMPANY-NAME, SALES, NET-PROFIT) and all associated context information (e.g., `currency`, `scale-factor`, `date-conventions`). Receivers must specify a similar context definition which will contain a list of the kinds of data that the receiver would be interested in seeing and the associated context information. Typically it is expected that a given receiver's context definition will be similar, if not identical, to the context definition of their favorite data source (usually their local data services). In the example shown in Section 2, the receiver's context was chosen to be identical to the Worldscope database.

Integral to the context definitions is what we call a *shared ontology*. The shared ontology essentially provides a standard vocabulary in which to express the context definitions. The vocabulary consists of two parts. The first is a standardized list of the different kinds of data that could be represented in a given database. In our current implementation the shared ontology is intended to cover the different kinds of data that might appear in a financial application. The second important functionality of the ontology is to describe the contextual information about the data. In Section 4 we will describe in more detail exactly how this is implemented.

The *context mediator* intercepts all requests for data made by the receiver, it translates the query from the receiver to the source context, poses the query to the source database capturing the data returned, and finally the data is converted into receiver context. The final stage of converting the data into the receiver context is perhaps the most complex, and involves consulting a library of conversion functions which enables the context mediator to convert from one context (e.g., currency or date conventions) to another.

3.2 Context Management

The Context Management Application allows the user to interact with the various source and receiver context definitions. The context manager provides the following services to the user:

1) Retrieval of the context of a given node (i.e., source or receiver.)
2) Updating of the context information for a pre-existing node.
3) Entering the context for a node which is new to the network.
4) Comparing contexts for two different nodes to produce a list of potential conflicts.
5) Comparing individual attribute from the same or different nodes to produce a list of potential conflicts.
6) Browsing of the shared ontology to learn more about the domain (e.g., financial) and potential sources of useful information.

As will become clear, several of the above services rely on the same underlying mechanism as is used by the context mediator in providing for semantic interoperability. In particular, the comparing of contexts to see the potential conflicts is exactly what is done as soon as a receiver chooses a particular data source. In order to determine which conflicts actually occur it is necessary to examine the query used. In the next section we will describe in detail how we implement many of the high level components shown in Figure 7.

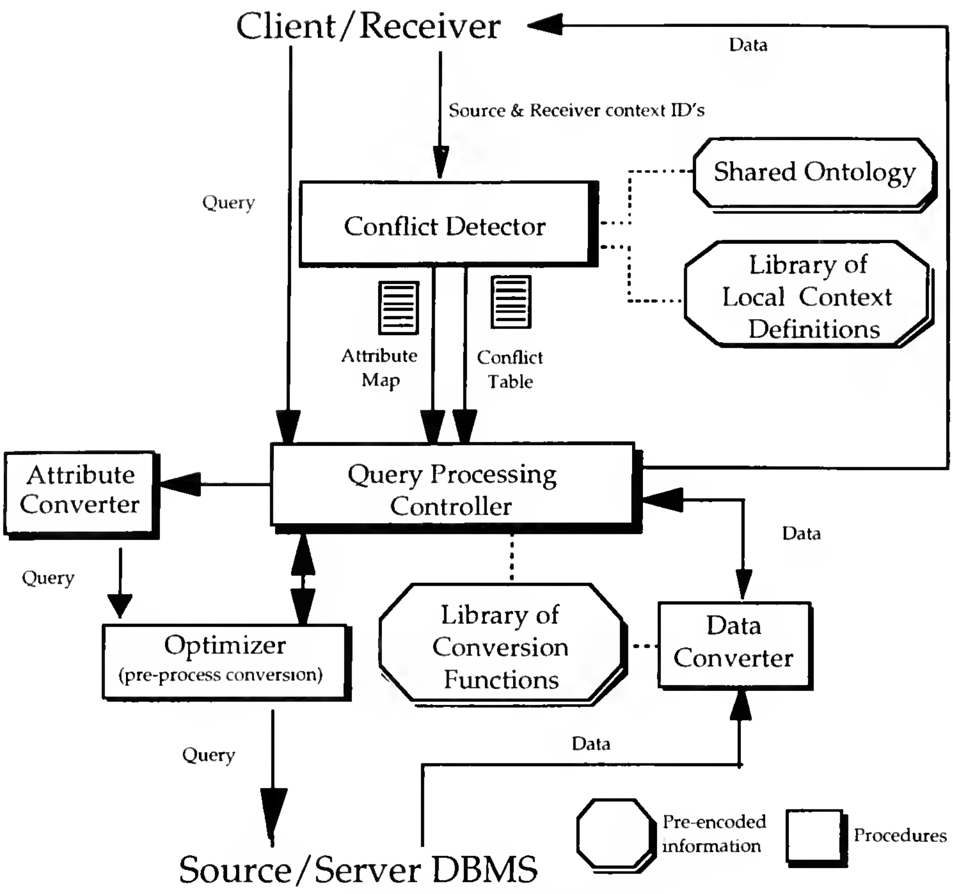


Figure 8: Context Mediator

4. CIN Implementation

At the core of the Context Interchange Network is the ability to represent context knowledge, perform conflict detection, and provide context transforming query processing. These capabilities are depicted in Figure 8.

The context representation is provided by the shared ontology (i.e., the set of common concepts) and the local contexts (i.e., the definition of a node's context and the mapping of that context to concepts in the shared ontology). Conflict detection is determined by the *conflict detector* which provides as output the *attribute map* and the *conflict table.*

The context-based query processing is controlled by the *query processing controller* that makes requests for the attribute map and the conflict table and accesses and applies the necessary conversion. The controller calls on the *attribute converter, data converter,* and *optimizer* to provide data to the receiver in the required context. In the remainder of this section we examine the key functionality of the CIN, context representation, conflict detection, and query processing.

4.1 Context Knowledge Representation

The crux of our effort to do Context Interchange involves our specification of what constitutes "context knowledge" and how it is represented and made available to the context mediator for conflict detection and transformation. This specification determines both the power and the inherent limitations of our current approach. As a starting point for this discussion consider Figure 9 which shows the source and receiver contexts from the example in Section 2. Shown in this diagram is the local context for the receiver (i.e., whom prefers the Worldscope context), the local context for the data source (i.e., the context for Disclosure), a shared ontology of concepts, and the mappings between the local context and concepts in the shared ontology. Each of these components of context knowledge representation will be described in this section.

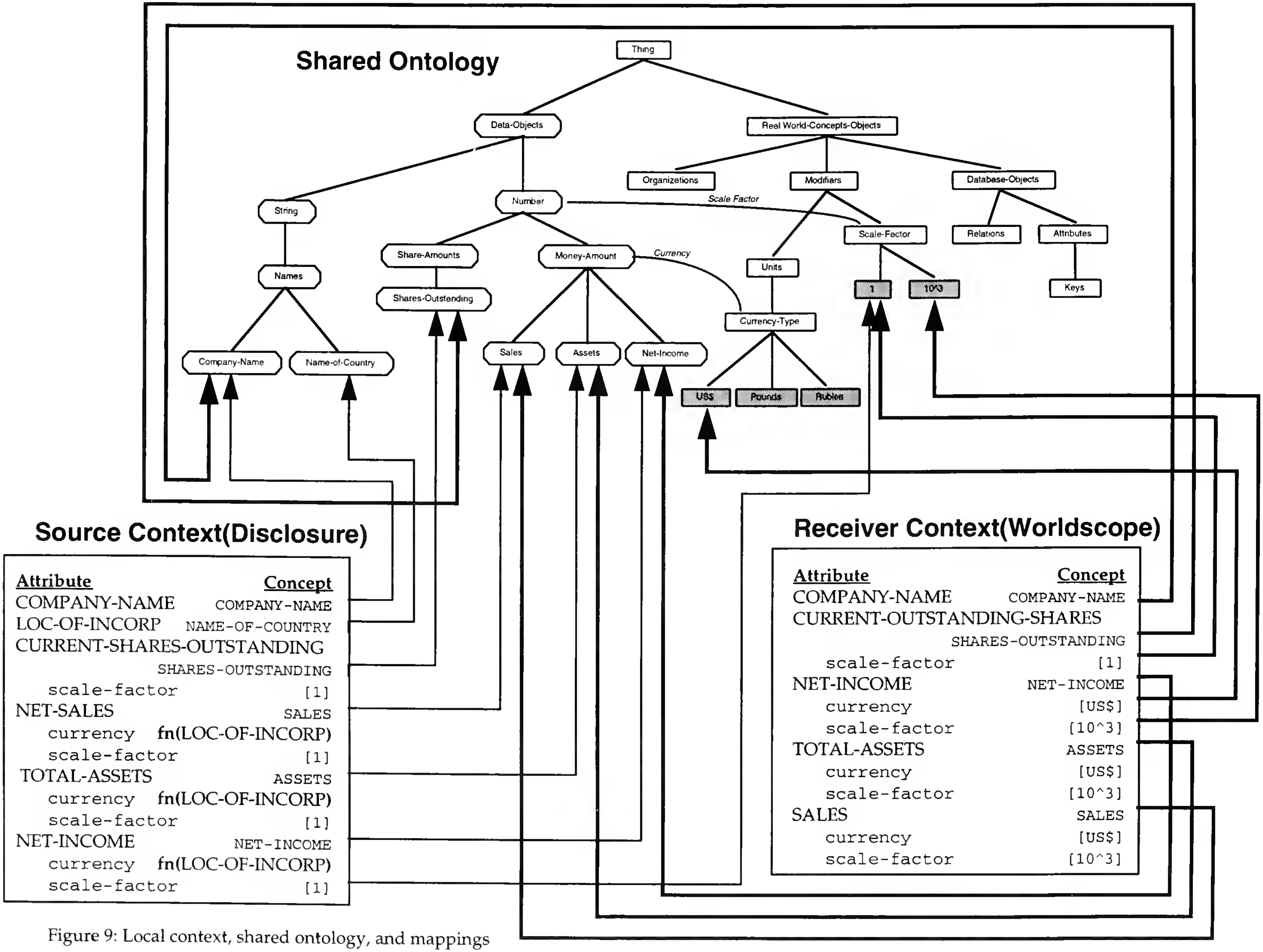
4.1.1 Defining Local Contexts

In our current implementation it is possible to attach contextual information only to the fields or attributes of a given node (i.e., source or receiver's view). The form of contextual information that we may encode is meta-data and is asserted by stating values for *meta-attributes*. An example of meta-data that might be found in traditional database dictionaries is whether the number for SALES is stored in the source data as a string, a float, or an integer. However, we are more concerned with semantic information that is not found in traditional data dictionaries. For example, the interpretation of SALES (see Figure 9) that comes from knowing its `scale-factor` or `currency`. In addition there are more complex types of meta-data, for example whether the value given in a data source for NET-INCOME includes or excludes extra-ordinary expenses (i.e. one-time costs like refinancing a pension). We make such assertions by giving values for meta-attributes on certain attributes. For example, in the receiver context shown in Figure 9, the NET-INCOME field has a value of [US$] for the meta-attribute `currency` which means that for the receiver the numbers that appear in the NET-INCOME field should represent numbers of US dollars. In Section 4.1.4 we examine the range of approaches for assigning meta-attribute values. In the next two subsections we examine the role and structure of the shared ontology and the mapping between the local context and the ontology.

4.1.2 The Shared Ontology

The shared ontology provides several important representations. First, the common concepts are represented so that disparate local contexts can be mapped to these concepts and context comparison may be done in a common language. Secondly, the ontology provides a representation for meta-attributes (i.e., shown as links). Finally, the shared ontology provides the identity for common conversion functions[3] that may be used to transform context differences. The specific shared ontology constructed for this implementation is

[3] The ability to represent both common conversion functions and those that may be specific to a given context is considered in this implementation. Local conversion functions may be identified in local context definitions. For simplicity, conversion function representation is not shown in the diagram and will not be covered in this paper.



Source Context(Disclosure)

Attribute	Concept
COMPANY-NAME	COMPANY-NAME
LOC-OF-INCORP	NAME-OF-COUNTRY
CURRENT-SHARES-OUTSTANDING	SHARES-OUTSTANDING
scale-factor	[1]
NET-SALES	SALES
currency	fn(LOC-OF-INCORP)
scale-factor	[1]
TOTAL-ASSETS	ASSETS
currency	fn(LOC-OF-INCORP)
scale-factor	[1]
NET-INCOME	NET-INCOME
currency	fn(LOC-OF-INCORP)
scale-factor	[1]

Receiver Context(Worldscope)

Attribute	Concept
COMPANY-NAME	COMPANY-NAME
CURRENT-OUTSTANDING-SHARES	SHARES-OUTSTANDING
scale-factor	[1]
NET-INCOME	NET-INCOME
currency	[US$]
scale-factor	[10^3]
TOTAL-ASSETS	ASSETS
currency	[US$]
scale-factor	[10^3]
SALES	SALES
currency	[US$]
scale-factor	[10^3]

Figure 9: Local context, shared ontology, and mappings

a financial reporting ontology and was developed based on the examination of more than eight financial data sources but with particular emphasis on the information represented in Disclosure, Worldscope and Compustat. It is important to note that the CIN is in no way limited to the financial domain. It may be used in any domain where there is sufficient context knowledge to provide a shared ontology and local context definitions.

Because of the large number of concepts that might be associated with a given domain, the shared ontology is defined as a taxonomy of concepts. In addition, because the shared ontology is designed for use in context interchange it is very useful to make the distinction between Data Objects and Real World Concepts and Objects. The ovals on left side of the tree (as shown in Figure 9) represent classes of data objects (e.g., SALES, COMPANY-NAME, or NAME-OF-COUNTRY) and the rectangles on the right side of the tree the represent classes of mostly abstract entities (e.g., MODIFIERS, ORGANIZATIONS, RELATIONS, SCALE-FACTORS). As shown in Figure 9 and described later, the left hand side provides the set of concepts for mapping the attributes defined in the local context's, while the right-hand-side provides concepts for defining meta-attributes and their values. Links between the data objects and the real world concepts are used to represent meta-attributes and can be read as " NUMBER has a meta-attribute of Scale Factor"

We have found this approach to structuring the shared ontology beneficial in both simplifying its development and using it for context interchange. One persistent and frequently confusing aspect of this dichotomy arises from the question of what information belongs on which side of the taxonomy, the right or the left? Fortunately this is not an either/or decision and the answer can be both, although it will not often be so. An example when this might happen involves the use of currency types. Clearly currency is an appropriate meta-attribute and thus we need to represent the different currencies as "real world concepts" and thus belonging to the right side of the tree. However, there may also exist a financial database which contains among other things a field dedicated to holding the names of currencies. In order to encode a context definition for this database it would be necessary to assign a concept to this attribute and a perfectly appropriate concept might be CURRENCY-NAME (subclass of NAME). Thus in this example we have nodes on both sides of the taxonomy with "currency" as their key descriptive element. However, it is important to note that these two nodes do not duplicate the same information, the CURRENCY node on the right hand side represents the set of abstract concepts which are the currencies of international trade which the node CURRENCY-NAME on the left hand side represents the set of data objects which are the names of the currencies of international trade. The distinction is subtle but important. A similar example might be that of the concept COMPANY which was the set of companies and COMPANY-NAME which is the set of company names.

This distinction between objects and their representations is defensible and indeed desirable on its own merits. A frequent problem in knowledge representation is the careless mixing of representations with objects. Some data objects such as the number of employees or net profit do not have easily identifiable objects which they represent and some objects such as companies or people to not easily convert to unambiguous representations. In addition while company names may have maximum lengths or punctuation conventions, companies do not, and while companies may have presidents or net incomes, company names certainly do not. While this distinction may seem clear in theory, in practice in large taxonomies it is easy to get hopelessly confused about what exactly certain named concepts exactly represent [Woods, 1985].

Another major piece of the shared ontology involves the relation of meta-attributes to the concept taxonomy. Meta-attributes are similar to binary predicates and relate two classes of things. From Figure 9, one can see that on the concept NUMBER the meta-attribute scale-factor is defined while on MONEY-AMOUNTS the meta-attributes currency is specified. Values can be specified for meta-attributes attached to a given concept for all the meta-attributes defined on that concept or on any concept which is an ancestor of that concept. Thus looking at the diagram we can see that the concept SHARE-AMOUNTS could specify a value for the meta-attributes scale-factor while the concept NET-INCOME could specify a value for the meta-attributes scale-factor and currency. It is important to note that while in theory we could specify a meta-attribute value anywhere from the root concept to the leaf, in our implementation we only allow leaf nodes to attach to attributes and specify the values of meta-attributes.

4.1.3 Mapping Local Context to the Shared Ontology

The local context definition contains attributes and meta-attributes that must be mapped to concepts in the shared ontology. The local context encodes what kind of information is contained in the given field, these correspond to concepts in the data object portion of the shared ontology. For example, the attribute called CURRENT-SHARES-OUTSTANDING in the source can be paired with the attribute CURRENT-OUTSTANDING-SHARES in the receiver because they have been linked to the same concept, SHARES-OUTSTANDING.

The addition of a new node into the Context Interchange Network would require that each attribute be paired with a concept in the shared ontology.[4] For each attribute they would begin at the root node and traverse a path to the appropriate leaf node in the tree, note that only the leaf nodes are valid concepts to attach to attributes. The principle guiding the design of this network of concepts is that the traversal process should be user friendly. This means that at each level there should be a reasonably small number of branches (i.e., less than five) and that the choices are sufficiently distinct that there is only a small probability of going down the "wrong" path for more than a couple levels. At

[4] In the current implementation we do not allow new concepts or links to be introduced to the shared ontology by a node. Later implementations will consider how to make the ontology more extensible for new concepts and meta-attributes.

any time, a user may consult the associated Real World Concepts to help clarify the meaning of a data object.

The meta-attributes in the local context are defined by the links to the real world objects. These links may be applied anywhere in the taxonomy and thus the meta-attributes for a concept are the collection of links for it and the parent concepts.

4.1.4 Assignment of Meta-Attribute Values

In this section we examine the three categories of meta-attribute value assignments and present those that are supported in the current implementation. The three categories are:

Independent: The value of such a meta-attribute is a member of some class of objects, and is known at the time the context is entered. For example, all the SALES figures for the receiver shown in Figure 9 are represented thousands of US dollars.

Closed Data Dependency: In this case the value of a meta-attribute attached to a given attribute is dependent on the value of data in another attribute(s) in the same record. For instance, the currency value for NET-SALES, TOTAL-ASSETS, and NET-INCOME for the source in Figure 9 is determined by the location of the incorporation of the company (e.g., Daimler Benz of Germany has net sales in German marks, Sony of Japan has net sales in Japanese Yen).

Open Data Dependency: Same as immediately above except that the value of the meta-attribute might be dependent on the value of data associated with an attribute(s) which may be in another record or sets of records (e.g., different relations in the same or other sources). As an example consider the case above except that the country of the corporate headquarters is not included as a field in the same record with net sales but instead is located in some other source[5]

The extent to which we support these three mechanisms in this implementation is as follows:

Independent: Fully supported.

Closed Data Dependency: Partially supported. Although we support this form of meta-attribute, we limit for now the nature of the relationship between the meta-attribute and the data which it is dependent upon. The user is allowed to

[5] Note that we do not need to include the case of a meta-attribute dependent on the value of another meta-attribute since either the original meta-attribute is a constant or is dependent on some attribute. Thus the value of the second meta-attribute is either a function of a constant and thus a constant or the composition of two functions applied to a piece of data and thus constituting some form of data dependency. In this analysis we ignore the possibility of a cyclic dependency of meta-attributes.

state rules of the form: if ATTRIBUTE relation [constant] then set meta-attribute equal to constant. The relation can be any of the more common binary predicates: equal, not equal, greater than, less than, etc. Thus for example a valid rule would be, if the field COUNTRY[6] is equal to "Germany" then the currency meta-attribute associated with NET-PROFIT would be set to German-marks. Later we plan to loosen this constraint on expressiveness.

Open Data Dependency: Not currently supported. Given the notion of linking one piece of data to another it is not difficult, conceptually, to expand that to linkages outside the current record. However, from a pragmatic point of view once one allows external linkages the number of different kinds of links grows from a handful to being almost unbounded. Potentially one can conceive of a number of external linkages that could be determined automatically although it is not immediately apparent how comprehensive or useful this list would be. It is clear, however, that in many cases it would be incumbent upon the user to identify and define which linkages they were interested in using, and while this certainly is not intractable it is highly ambitious.

As an example of a closed data dependency we direct the reader again back to Figure 9. All the currencies in the Disclosure database are in the base currency of the company, thus all financial figures for French companies are in Francs, for Germany companies they are in Marks, etc. The way that this is actually encoded is by attaching a number of meta-attribute settings to each attribute each with a closed data dependency. Thus for NET-SALES, for example, we would have the following rules:

currency=[US$]	if	LOC-OF-INCORP="USA"
currency=[Marks]	if	LOC-OF-INCORP="Germany"
currency=[Francs]	if	LOC-OF-INCORP="France"
currency=[Pounds]	if	LOC-OF-INCORP="UK"

The taxonomic representation does allow for the user to view and make assignments for related families of concepts. Thus if the user knows that in a given database all of the money amount figures are in US dollars or that all dates are in European format then it will be possible to state this fact about the high level concept and have it be inherited by all its descendant attributes. Similarly when viewing the context for a data source or data receiver the system will report meta-attribute assignments at the highest level that it can. Thus it may be that when entering the context of a given data source the user will individually assert that all the financial attributes are in US dollars because she does not see the pattern. However, when the context is displayed it will show that all MONEY-AMOUNTS have the meta-attribute currency set to [US$].

4.1.5 Underlying Representation System: LOOM

The system that we use for implementing the above taxonomy is a general purpose taxonomic knowledge representation system called LOOM [MacGregor,

[6] This would mean country of location of the headquarters of the company.

1991], which is implemented in LISP. LOOM is designed to allow for the representation of a great variety of concepts, entities, and relationships between them. However, we use only a small fraction of the power that LOOM provides in our current implementation. Our reasoning in choosing such a vast system was to have the maximal expressive capabilities at our disposal and then to later redesign the system for efficiency by removing all the features that were not important or that we did not use.

In LOOM and the description logics view of things, the world consists of a set of entities which can be either abstract or concrete and consists of all the things that the user is interested in reasoning about. These entities are then broken up into various subsets which are called concepts. Thus in knowledge representation systems it is crucial when reasoning about concepts to also be aware of what exactly is the set of entities that the concept denotes. The last piece of the LOOM view of the world is what are called *roles*. Roles are two place predicates and thus are true for some set of ordered pairs of entities and false for all other pairs. In our implementation meta-attributes are implemented in LOOM as roles.

The following is a sample of LOOM code that is used to represent the local context for the receiver in Figure 9 (i.e., the Worldscope context):

```
(defconcept WrldScp-Annual-Fin-Attributes :is-primitive Db-Attribute)

(tellm (:about WrldScpAF Worldscope-Relation
            (:filled-by has-attribute
                       'WrldScpAF-CompanyName
                       'WrldScpAF-LatestAnnualDate
                       'WrldScpAF-CurrentOutstandingShares
                       'WrldScpAF-NetIncome
                       'WrldScpAF-Sales
                       'WrldScpAF-TotalAssets)))

(defconcept WrldScpAF-CompanyName :is-primitive Company-Name)

(defconcept WrldScpAF-LatestAnnualDate :is-primitive Latest-Annual-Date)

(defconcept WrldScpAF-CurrentOutstandingShares :is-primitive
           Shares-Outstanding)

(defconcept WrldScpAF-NetIncome :is-primitive
  (:and Net-Income
      (:filled-by scale-factor times-1000)))

(defconcept WrldScpAF-Sales :is-primitive
  (:and Sales
      (:filled-by scale-factor times-1000)))
```

It is important to note that some of the local context is inherited from parent concepts. For example, `currency = [US$]` is inherited for Worldscope money amounts (e.g., SALES).

4.2 Conflict Detection

Although LOOM provides a great many services to the user, unfortunately conflict detection between differing context definitions is not among them, therefore we have built a LISP program that interacts with LOOM to perform conflict detection and produces the attribute map and the conflict table as shown in Figure 8 and described in this section.

4.2.1 Attribute Maps

The attribute map is essentially a table that pairs attributes from one node with attributes from another node. For the example of Section 2 the attribute map would be as follows

Worldscope	Disclosure
NET-INCOME	NET-INCOME
SALES	NET-SALES
CURRENT-OUTSTANDING-SHARES	CURRENT-SHARES-OUTSTANDING

The manner in which this table is generated is fairly simply. After the application has indicated which receiver attributes are of interest, then the conflict detector goes through each receiver attribute one at a time. It looks at the concept that is associated with each attribute and checks to see if there is an attribute in the source context definition that is also associated with that concept, if there is then the attributes match and if not then there is no attribute that corresponds to the receiver attribute.

4.2.2 Conflict Tables

The generation of the conflict table is somewhat more complex than for the attribute map. The conflict table consists of several entries, the attribute name (in the source context) on which the conflict may appear, the name of the meta-attribute on which the context definitions differ, the source value of the meta-attribute, the receiver value of the meta-attribute, the name of a function that could convert the data appropriately, and a predicate which determines if for a particular piece of data whether there is a conflict. Thus again referring back to the example from Section 2 and Figure 9 we might have the following excerpt from the conflict table:

Attribute	Meta-Attr	Src	Rcv	Conv-fn	Predicate
NET-INCOME	currency	[Marks]	[US$]	currency-cvt	LOC-OF-INCORP="Germany"
NET-INCOME	currency	[Pounds]	[US$]	currency-cvt	LOC-OF-INCORP="England"
NET-INCOME	currency	[Francs]	[US$]	currency-cvt	LOC-OF-INCORP="France"
NET-INCOME	scale	10^3	1	scale-cvt	true
SALES	scale	10^3	1	scale-cvt	true

Note that there would be many more entries for other different currencies however they have been omitted to provide a reasonable size example. In other words the table above states that for the attribute NET-INCOME the currency used is dependent on another field in the same record, LOC-OF-INCORP. The last line in the table simply states that the scale factor on the SALES and NET-INCOME attribute is 10^3 in the source database and only 1 for the receiver and that those scale factor settings are not dependent on anything else and thus the predicate is always true. For the NET-INCOME requested on Daimler-Benz of Germany, the first and fourth lines of the table are used.

Fortunately the generation of this table, in principle, is also reasonably simple. For any two attributes which have been paired in a attribute map we go to the concept in the taxonomy to which they have been linked and compare how the source and receiver specify their meta-attribute settings. If the settings disagree and the settings of the values are "independent" as discussed in Section 4.1.4 then we place a simple entry in the table (similar to the last line in the table above). If the setting of the meta-attribute value uses a closed data dependency then the condition is placed in the predicate line.

4.3 Context-Based Query Processing

In this section we describe how a query from the receiver are resolved in the receiver's context. The components of this process are shown in the bottom half of Figure 8 beginning with the Query Processing Controller .

4.3.1 Query Processing Controller (QPC)

This module controls all other components of the query processing engine as well as provides an interface to external data sources and receivers. The controller maintains the state of execution of a query. This state includes node names, passwords, and execution states. It also retrieves essential information from the conflict detector regarding conflicts between nodes in the form of a conflict table and an attribute map.

4.3.2 Query Optimization

The optimizer is a Selinger-style [Selinger, 1979] query optimizer (implemented in C++ with tools from [Corporation, 1993]) that generates an optimal query execution plan given an SQL statement, a conflict table and an attribute map. Queries that are directed through the QPC are decomposed into the primitive operators (e.g. selection filters, joins, projections). Furthermore, conversion operators are generated for each conflict in the conflict table that relates to the active query. The context-based query optimizer determines the location of selections and joins (i.e. conventional query optimization), and incorporates conversion filters into the query plan (i.e. conversion-based optimization). Each operator, has two distinct sets of properties, physical and logical properties. Physical properties include the cost of the operator and the applicable algorithms to implement the operators. Logical properties describe the logical effect of the operator on a data set.

Conversion operators are inserted into the plan to assure logical consistency (e.g., join attributes must be in the same context, selection constants and attributes must have the same context and output data must be in the receiver context).

Hence, if a conflict exists on the attribute SALES then no operator can be applied on SALES without a conversion of SALES data. The identity of conversion functions are provided in the conflict table. Also identified is the existence of inverse conversion function. A conversion function (*Fn*) (i.e., the source-to-receiver conversion function) has an inverse (Fn^{-1}) if the inverse function (Fn^{-1}) can be used to transform a predicate in the receiver's query into the context expected by the source (i.e., receiver-to-source conversion). The presence of inverse conversions will be shown to be very important in the optimization of process.

The optimal execution is generated by exhausting all possible combinations of *operator nodes* in the plan. All plans that are not logically consistent with the original query are pruned. Appropriate physical algorithms are attempted for each operator of the plan and the resulting costs are calculated. The optimal plan is selected from the possible plans on a cost minimizing basis.

The optimizer is implemented in C++ and is extensible to handle additional operators and changes to cost functions. Each node in the optimizer can have multiple algorithms associated with it as well as separate costs based on those algorithms.

An illustrative example, based on the scenario described in Section 2, of query optimization is described below. The user requests current shares outstanding, sales and net income information for the German company Daimler-Benz AG from the Disclosure database generating the following SQL statement:

Select COMPANY-NAME, CURRENT-SHARES-OUTSTANDING, SALES,
NET-INCOME **From** Disclosure
Where COMPANY-NAME="DAIMLER-BENZ AG" **And** DATE=12/31/93

Both the NET-INCOME and SALES attributes are represented in German Marks in the Disclosure database and they also have a scale factor of 1000. The data receiver prefers her sales and net income figures to be in US Dollars with a scale factor of 1. Furthermore, the receiver and source differ on the naming convention used for company names (i.e. receiver: "DAIMLER-BENZ AG", source: "DAIMLER BENZ CORP").

Figure 10 shows two of the query trees generated by the optimizer. The trees are non-dependent sequential trees with data flowing from the bottom (i.e. data at the source) to the top (i.e. output to the receiver). Figure 10(a) depicts a suboptimal execution plan for the query described above. The **Relation** node signifies the relation Disclosure on disk. The **File Scan** node retrieves relevant attributes from Disclosure. The relevant attributes include: all attributes present in the output list , all attributes in the predicate of the SQL statement (e.g., DATE) as well as any attributes required for a conversion function (e.g., LOC-OF-INCORP to determine the currency). The tree then incorporates three **Convert** nodes, for the attributes COMPANY-NAME, NET-INCOME and SALES, that ensure that output to the receiver has been converted to her context. The **Select** node is an operator that filters out the records for the company Daimler-Benz and for the 1993 fiscal year. The **Output** node is the final node in the query tree and signifies the presentation of the results of the query to the user. Hence, the tree in Figure 10(a) describes the process of reading all data from the Disclosure relation, converting all the data and then filtering out those data of interest to the receiver.

In Figure 10(b) shows an optimal plan. At the bottom of the tree is a **Relation** node indicating the Disclosure relation as in Figure 10(a). However, the **Select** node precedes the **File Scan** node in this query tree and contains an inverse conversion function applied to the value of the COMPANY-NAME attribute (i.e., converting to the Disclosure context). The selection filter can now be applied as the data is being scanned by the **File Scan** operation at minimal extra cost (as compared to a cost proportional to the size of the relation). The COMPANY-NAME must be converted from the source context to the receiver context before output to the receiver. The query tree in Figure 10(b) clearly has lower cost than the tree in Figure 10(a) because the number of records on which conversions are performed is significantly reduced by placing the **Select** node before the conversions.

More complex queries are handled in a similar manner. Joins between two sources necessitate transformations to the context of either source or another common context before join attributes can be compared. All three techniques are attempted and the most efficient is chosen based on the size of the two source relations and the attributes required in the query output.

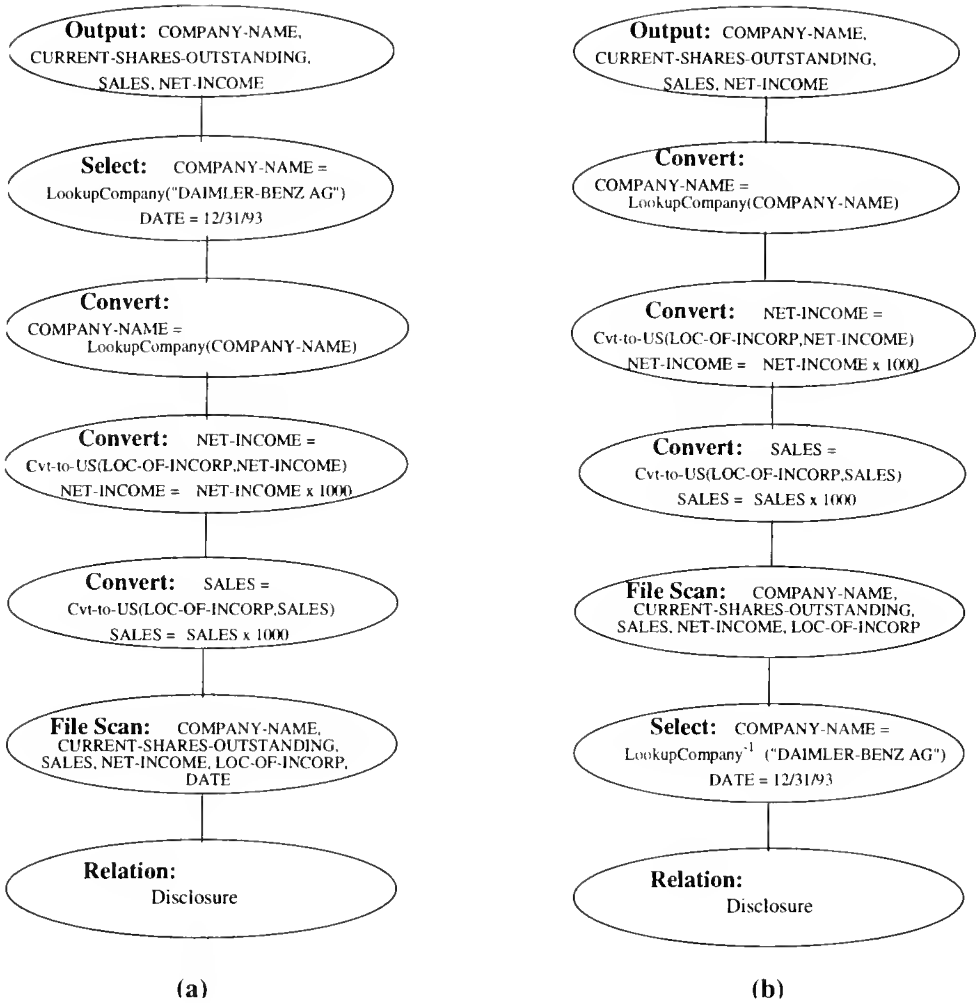

Output: COMPANY-NAME, CURRENT-SHARES-OUTSTANDING, SALES, NET-INCOME
Select: COMPANY-NAME = LookupCompany("DAIMLER-BENZ AG") DATE = 12/31/93
Convert: COMPANY-NAME = LookupCompany(COMPANY-NAME)
Convert: NET-INCOME = Cvt-to-US(LOC-OF-INCORP,NET-INCOME) NET-INCOME = NET-INCOME x 1000
Convert: SALES = Cvt-to-US(LOC-OF-INCORP,SALES) SALES = SALES x 1000
File Scan: COMPANY-NAME, CURRENT-SHARES-OUTSTANDING, SALES, NET-INCOME, LOC-OF-INCORP, DATE
Relation: Disclosure
(a)
Output: COMPANY-NAME, CURRENT-SHARES-OUTSTANDING, SALES, NET-INCOME
Convert: COMPANY-NAME = LookupCompany(COMPANY-NAME)
Convert: NET-INCOME = Cvt-to-US(LOC-OF-INCORP,NET-INCOME) NET-INCOME = NET-INCOME x 1000
Convert: SALES = Cvt-to-US(LOC-OF-INCORP,SALES) SALES = SALES x 1000
File Scan: COMPANY-NAME, CURRENT-SHARES-OUTSTANDING, SALES, NET-INCOME, LOC-OF-INCORP
Select: COMPANY-NAME = LookupCompany-1 ("DAIMLER-BENZ AG") DATE = 12/31/93
Relation: Disclosure
(b)

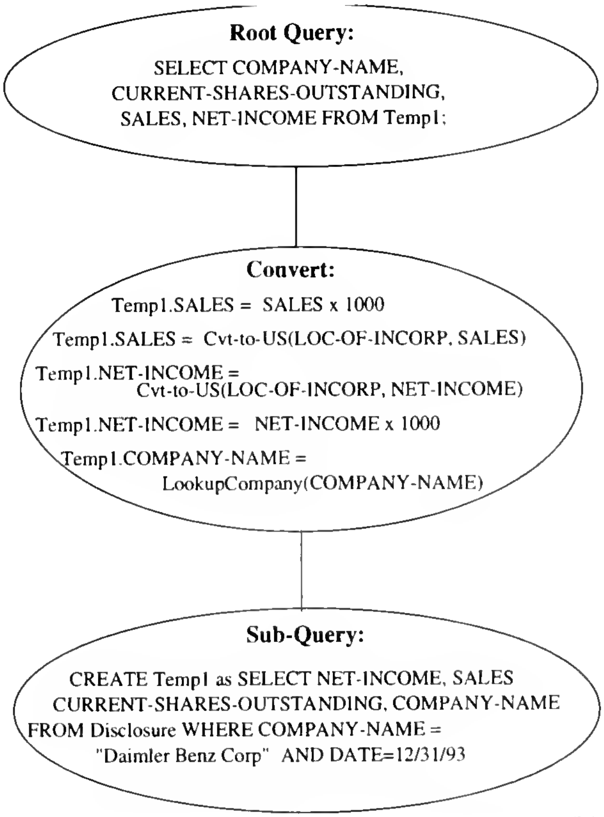
Figure 10: Possible Execution Plans (a) sub-optimal, (b) optimal.

4.3.3 Data Retrieval and Conversion

The execution plan representation in the optimizer is generic and does not correspond to a specific query language. In the data retrieval stage, the plan generated by the optimizer is transferred into SQL. Operator nodes below a **Convert** operator are transformed into an SQL subquery, and results are stored in temporary tables (see Figure 11). Conversion routines are applied to the temporary tables and subsequent queries are performed on the temporary tables. The final **Root Query** returns data to the receiver.

The context mediator supports three forms of conversion functions. These functions include: mathematical functions implemented in C++; lookup functions implemented as SQL queries; complex functions requiring interaction with the context repository.

Figure 11 shows the SQL execution plan generated from the query trees in Figure 10. Data flows from the bottom (the data source) to the top (data receiver) in the tree shown in Figure 11. The bottom node in Figure 11 is an SQL query to be sent to the Disclosure database, the results of which are stored in temporary storage, Temp1, under the control of the query processing controller. Note that the query has been modified by the conversion of the constant "DAIMLER-BENZ AG" to "DAIMLER BENZ CORP" through the use of the inverse conversion function *LookupCompany^{-1}()*. The conversions on COMPANY-NAME, SALES and NET-INCOME[7] are performed on data in Temp1. Finally, after all conflicts have been resolved an SQL query is applied to Temp1 to retrieve the attributes requested by the user.



```
SELECT COMPANY-NAME,
CURRENT-SHARES-OUTSTANDING,
SALES, NET-INCOME FROM Temp1;
```

```
Temp1.SALES =  SALES x 1000
Temp1.SALES = Cvt-to-US(LOC-OF-INCORP, SALES)
Temp1.NET-INCOME =
          Cvt-to-US(LOC-OF-INCORP, NET-INCOME)
Temp1.NET-INCOME =  NET-INCOME x 1000
Temp1.COMPANY-NAME =
          LookupCompany(COMPANY-NAME)
```

```
CREATE Temp1 as SELECT NET-INCOME, SALES
CURRENT-SHARES-OUTSTANDING, COMPANY-NAME
FROM Disclosure WHERE COMPANY-NAME =
     "Daimler Benz Corp"  AND DATE=12/31/93
```

Figure 11: Plan for optimal execution of Figure 10(b).

5 Middleware

The essential functionality of the context mediator is to detect context difference between source and receiver, transform queries accordingly, and apply all necessary conversions to the output.

Most applications, previously implemented and currently being implemented, did not have provisions for context mediation. An example of such an application is shown in Figure 12. This corresponds to the Data Extraction Application described in Section 2 with the Client Receiver Applications being

[7] Note that conversion of the scale factor in SALES and NET-INCOME could have been performed by adding the function to the target list of the subquery (i.e. CREATE Temp1 as SELECT NET-INCOME*1000, SALES *1000 ...). However, our current implementation does not take advantage of this feature in SQL .

implemented using Powerbuilder and the Source Database Management System being two Oracle servers, one holding the Worldscope database and the other holding the Disclosure database.

One of our goals is to be able to incorporate context mediator capabilities into such a situation in a non-intrusive way, that is, without requiring any change to the existing applications or the servers. Besides the obvious benefits of such an approach, that is eliminating the labor effort of converting existing software, it is often a practical necessity because either the receiver environment or source environment (sometimes both) may be outside the receiver's control (i.e., controlled by other organizations or unrelated parts of your own organization). Thus, changes many not be allowed.

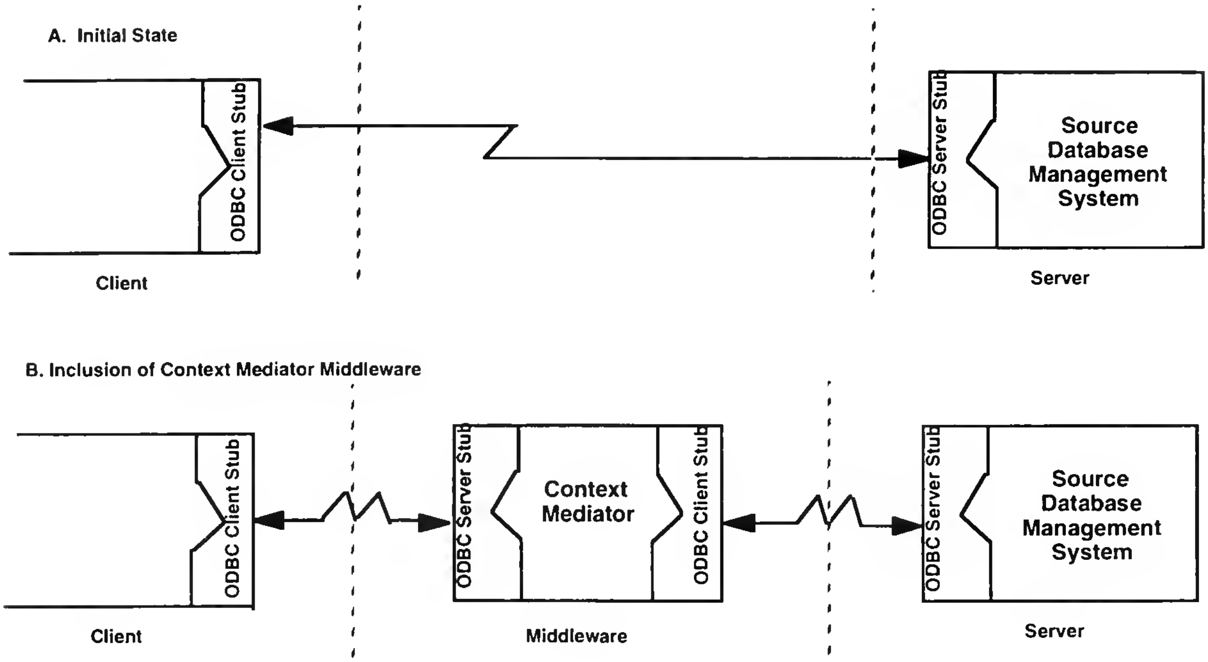


Figure 12: Context mediator middleware.

5.1 Three-Tier Middleware Structure

To provide distributed data access and to ensure compatibility with existing applications and storage systems we have elected to use a three-tiered middleware approach for the design of the context interchange network. (see Figure 12b). The three-tiered approach also allows a single context mediator to be simultaneously connected to multiple sources and receivers. The first tier is the client which may be any ODBC compliant application (Access, Excel, Custom applications). The context mediator, the second tier or middleware, performs conflict detection, query optimization and query execution. The third and final tier is the source database management system which must also be accessible through an ODBC driver. Connections between the multiple tiers are performed via the OEC ([Corporation, 1993]) RPC (remote procedure call) tools.

The context mediator has been implemented in C++ and Loom. The Loom portion handles the context repository and conflict detection mechanisms for the context mediator. The C++ portion includes the query parser, the conversion based optimizer and a query execution unit.

5.2 Open Database Connectivity (ODBC)

The context mediator engine is intended to provide seamless context conversion between applications and databases. Hence, the mediator uses the ODBC/API (Open Database Connectivity/Application Programming Interface) [Microsoft, 1994] which is an accepted industry standard. The ODBC/API includes a large amount of function specifications that fall under three main categories: (a) Connection Setup, (b) Query Submission and (c) Data Retrieval. Receiver context is determined during connection setup and is a function of the client application and the user. The query submission stage is essentially unaltered from any other application, however, internally, queries are redirected to the context mediator rather than the source specified by the user. Finally data returned in the retrieval stage will have been modified to comply with the receiver's context.

5.3 Legacy and other non-ODBC Sources

In reality, a tremendous amount of current data is in non-ODBC sources, often referred to as "legacy systems". These are non-ODBC for several possible reasons: (1) they pre-date modern relational database technology (e.g., IBM IMS) and/or based on some proprietary technology, (2) they are not designed to allow for direct database query access, either due to oversight, different goals, or as a form of protection, and only allow access through an interactive user interface (e.g., many online services), or (3) they have chosen to support another standard API other than ODBC. Often several of these reasons apply simultaneously.

To deal with these situations, we have used various commercial tools [Corporation, 1993] to develop ODB-adapter middleware. These are sometimes referred to as "wrappers" or "screen scrapers", we will use the term "connectivity middleware." Referring to Figure 14b, there is an additional layer of middleware between the context mediator and the source. The connectivity middleware presents an ODBC interface to the context mediator but presents a specialized interface to the source, as exemplified below.

The Worldscope data used in the example of Section 2 is actually available in several forms, we have used two forms: (1) a CD-ROM of quarterly updated data which was loaded into an Oracle relational database and (2) continually updated data is available as an online service which can only be accessed through an ASCII text menu selection interface (e.g., you are asked, "Please enter company name", a name such as "Daimler-Benz" is entered, and then a fixed format text report is displayed such as "NET INCOME 346,577, SALES 56,268,168" etc.)

The Worldscope connectivity middleware that we have constructed allows us to use either source in exactly the same manner, as ODBC sources, although there are differences in speed, cost, and updateness of the data.

5.4 Future needs of the Information SuperHighway

The topic of the "Information SuperHighway", or its current form as the Internet, has received considerable attention in government, business, academic, and media circles. Although a major point of interest has focused on the rapidly increasing number of users, sometimes estimated as 20 million and growing, an even more important issue is the millions of information resources that will be accessible.

Today, when people talk about "surfing the 'net", they are usually referring to use of the World Wide Web (WWW) through some user friendly interface, such as Mosaic or Netscape. This type of activity can be effective for casual usage and requires significant human intervention for navigation (i.e., locating the appropriate sources) and interpretation (i.e., reading and understanding the information found.)

On the other hand, let us consider the opportunity and challenges posed by exploiting these global information resources in an integrated manner. Let us assume that information from the various stock exchanges (possibly with a delayed transmission for regulatory purposes) and weather services around the world are accessible. (Note: They can be adapted for use by techniques described in Section 5.3 without requiring any change to the source.) You might want to know the current value of your international investments, which might require access to multiple exchanges both in the USA (e.g., NYSE, NASDAQ, etc.) and overseas (London, Tokyo, etc.) You might want to know where are the best ski conditions at resorts around the world. To manually access and interpret the numerous information sources relevant to these questions would rapidly become impractical. Furthermore, it is highly unlikely that all these sources would have the same context given their autonomous and geographically dispersed nature.

The three-tier middleware approach provides the opportunity to introduce context mediation services into such an environment in an incremental, evolutionary, and non-intrusive (or minimally intrusive) manner. Such services will be crucial to the effective, efficient, and correct use of information resources made available through the Information SuperHighway.

6. Conclusions

In this paper we have described an implementation of a Context Interchange Network to support semantic interoperability among heterogeneous sources and receivers. The CIN utilizes a declarative approach to context knowledge (i.e., rather than embedded in code, manuals, or the minds of users). It supports automatic detection of semantic conflicts and where possible the translation of the conflicts to provide the receiver with meaningful information (i.e., data in the receiver context). The CIN also supports the explication of data context; allowing for intelligent information browsing.

The development of this prototype has provided considerable insight into important research areas. First, there is a tremendous requirement for design methodologies and knowledge acquisition tools for the development and maintenance of shared ontologies and local context definitions. Our current ontology was developed to support the integration of eight financial data sources and as such was developed, for the most part, in a bottom-up approach. Data objects were constructed to allow for mapping of the three databases currently used in the implementation. Real world concepts were added to support meta-attributes. Some additional concepts were added to support understanding of data structures and further explication. This proved to be a very time consuming process without an apparent optimal strategy. Current effort are being spent on developing a user interface to support defining, browsing, and editing the local context knowledge, the shared ontology and mappings between the local context definitions and the shared ontology.

Second, the implementation needs to be extended to support more complex context knowledge and conflict reconciliation. The examples in the paper oversimplify the current capabilities but at the same time we are studying the need for the ability to better represent both derived (e.g., earnings-per-share) and combined (e.g., address, date) data and to reconcile differences between these types. Part of this process includes current efforts in developing a common theory for context definition, ontologies and databases.

Thirdly, the current query optimizer works mostly on compile-time conflict knowledge. As shown in [Siegel & Madnick, 1991], many of the context conflicts can only be determined at run-time, some of the capabilities to do run-time context checking have been incorporated but further analysis of the optimization process is needed to fully support this capability.

Finally, we are developing tools and methodologies to support system evolution and scalability (e.g., the use of multiple shared ontologies). These approaches will have significant impact on the ability for current and future Information Super Highways to deliver meaningful information exchange among multitudes of disparate sources and receivers.

7. References

[1] Corporation, O. E. (1993). *OEC Toolkit 1.1.* InOpen Environment Corporation.

[2] Goh, C., Madnick, S., & Siegel, M. (1994). *Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment.* In International and Knowledge Management (Ed.), Gaithersbury, Maryland

[3] MacGregor, R. (1991). Using a Description Classifier to Enhance Deductive Inference. *Proceedings of Seventh IEEE Conference on AI Applications*, (pp. 141-147) Miami, Florida.

[4] Microsoft (1994). *Microsoft's ODBC 2.0: Programmer's Reference and SDK Guide.* InMicrosoft Press.

[5] Selinger, P., et. al. (1979). Access Path Selection in a Relational Database Management System. *Proceedings of the 1979 ACM-SIGMOD International Conference on the Management of Data*, (pp. 23-34) Boston, MA.

[6] Siegel, M. & Madnick, S. (1991). A Metadata Approach to Resolving Semantic Conflicts. *Proceedings of the 17th International Conference on Very Large Databases*, Barcelona, Spain.

[7] Woods, W. A. (1985). What's in a Link: Foundations for Semantic Networks. In *Readings in Knowledge Representation.* Los Altos, California: Morgan Kaufmann Publiser's Inc.